Feature-Oriented Cache Designs

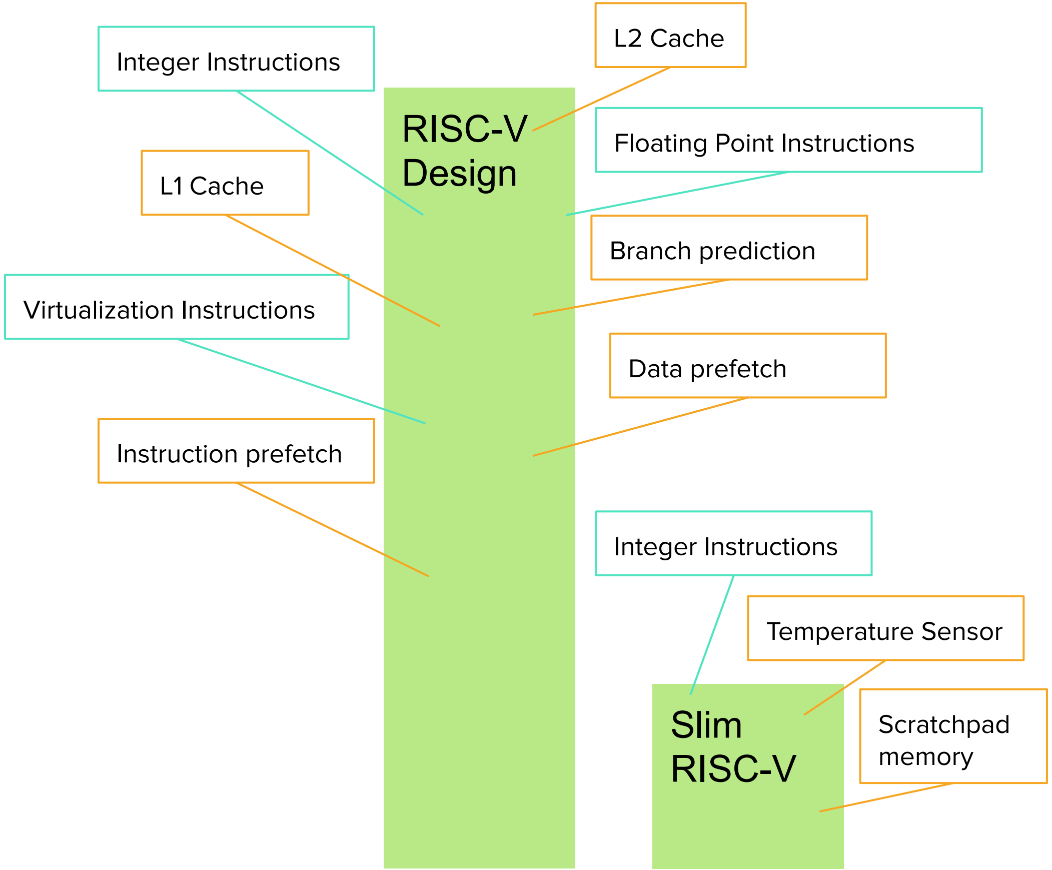
CARRV 2023

Justin Deters

Supported by NSF CISE award CNS-1763503 and SimpleRose

RISC-V

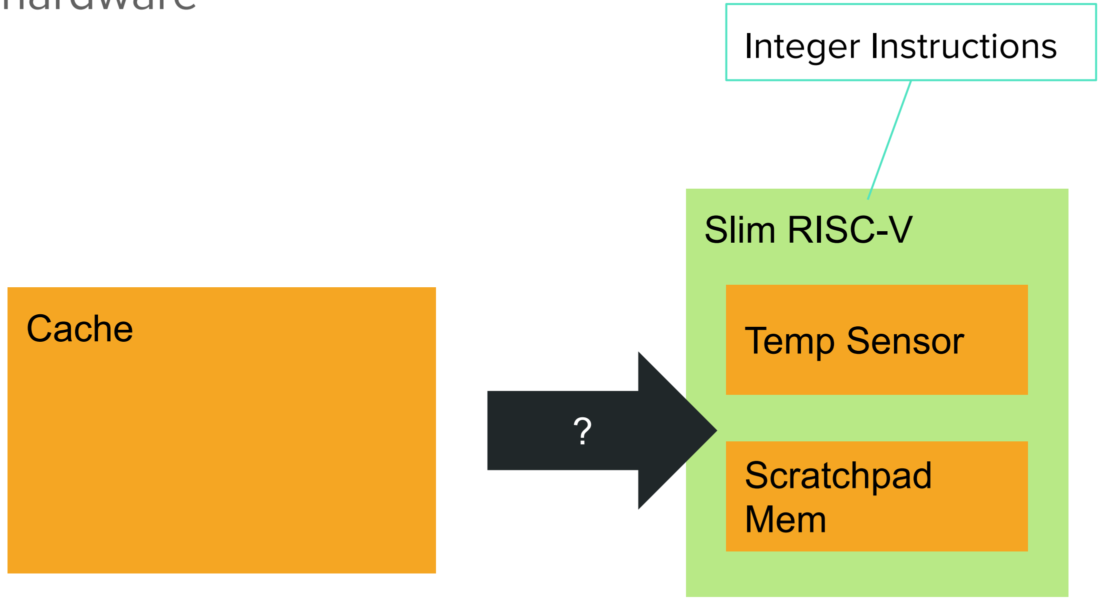
- The sea of hardware features is vast.
- Hennessy and Patterson introduced RISC-V[1].
    - Royalty-Free
    - Open Source
- Many characterizations
    - RocketChip, RISC-V Mini, BOOM, SERV, picorv32, SweRV, scr1



[1] J. Hennessy and D. Patterson. John Hennessy and David Patterson Deliver Turing Lecture at ISCA 2018. https://www.acm.org/hennessy-patterson-turing-lecture, 2018.

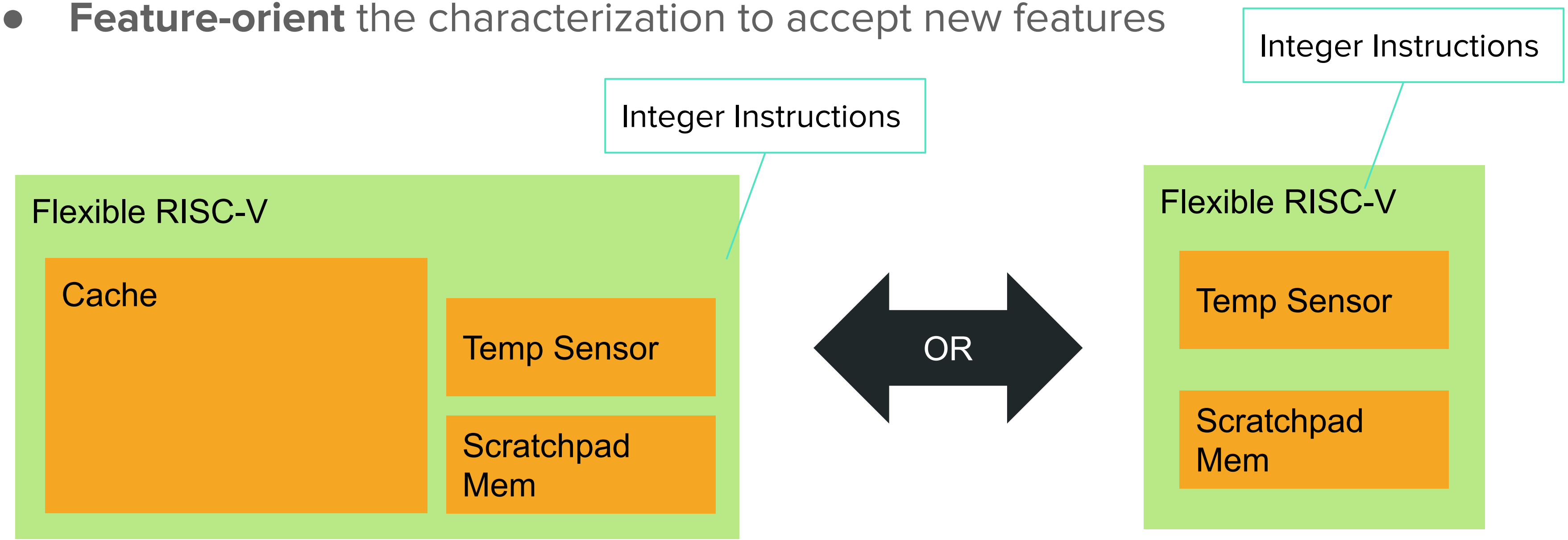
Adding a Cache

- How does this fit into slim RISC-V?
- Need new hardware



Flexible Characterizations

- Adaptable both qualitatively and quantitatively
- **Feature-orient** the characterization to accept new features


Integer Instructions

Flexible RISC-V

Cache

Temp Sensor

Scratchpad Mem

OR

Integer Instructions

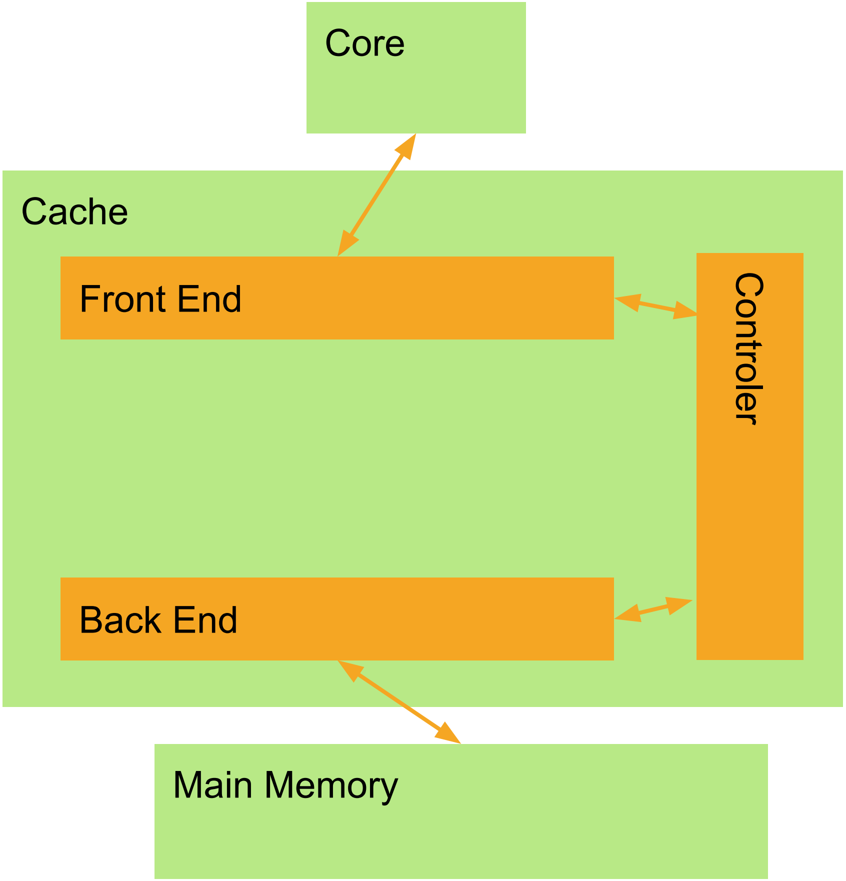
Flexible RISC-V

Temp Sensor

Scratchpad Mem

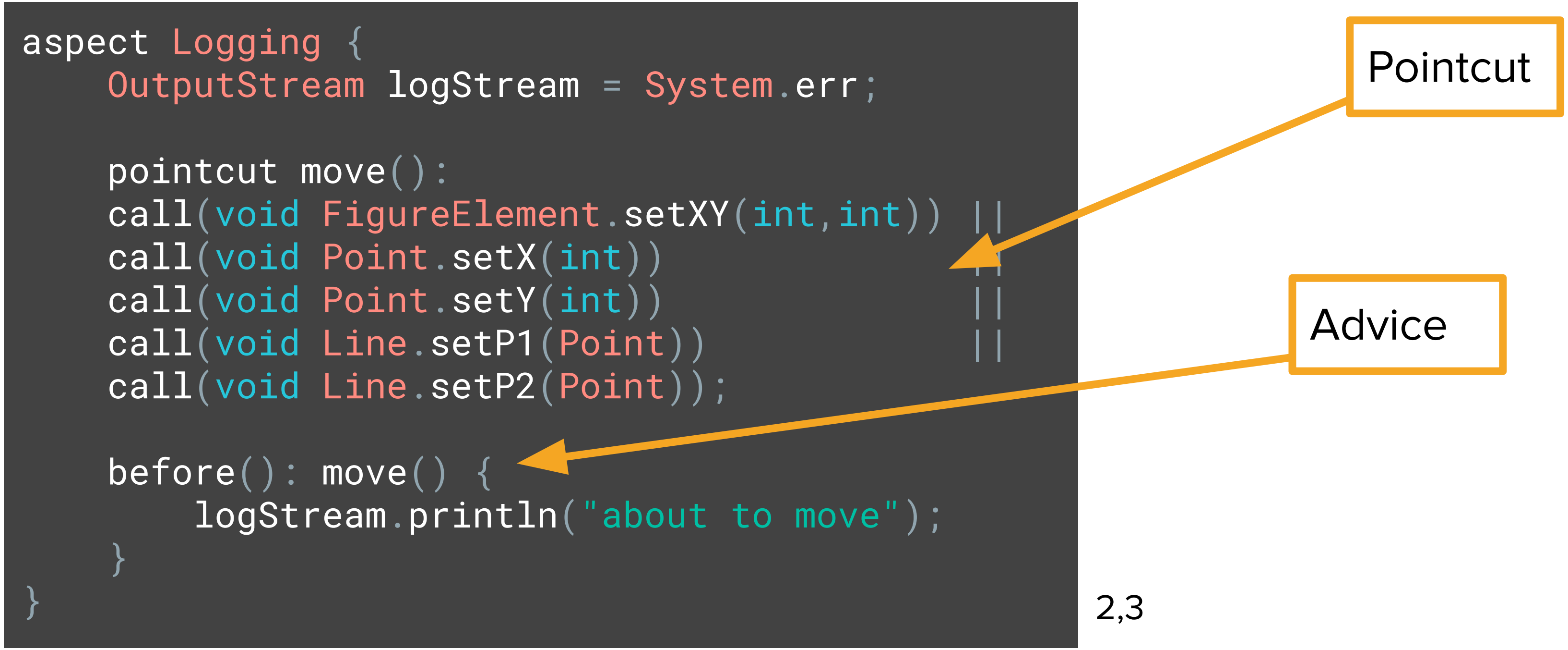
Caches

- Caches are ubiquitous in computing.
- Qualitatively
  - Write-Back vs Write-Through
  - Write-Allocate vs No-Write Allocate
  - Replacement Policy
  - Inclusion Policy
- Quantity
  - Cache Line Size
  - Number of Cache Lines
  - Number of cache levels



1. Feature-Oriented Finite-State Machines
2. Feature-Oriented Cache Designs
   a. Build on work from CARRV 2021
   b. Add rich type information

Aspect Oriented Programming[1]



```
aspect Logging {
    OutputStream logStream = System.err;

    pointcut move():
    call(void FigureElement.setXY(int,int)) ||
    call(void Point.setX(int))              ||
    call(void Point.setY(int))              ||
    call(void Line.setP1(Point))            ||
    call(void Line.setP2(Point));

    before(): move() {
        logStream.println("about to move");
    }
}
```

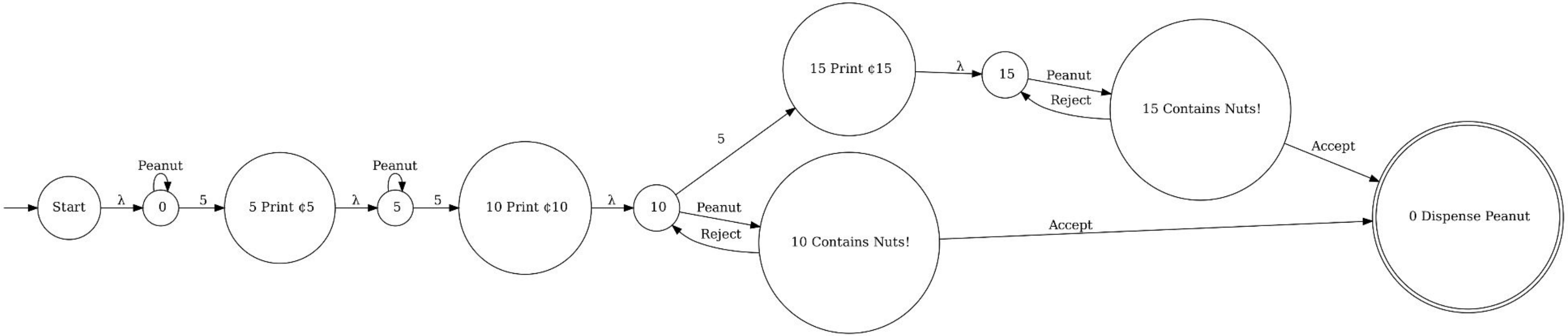
Pointcut

Advice

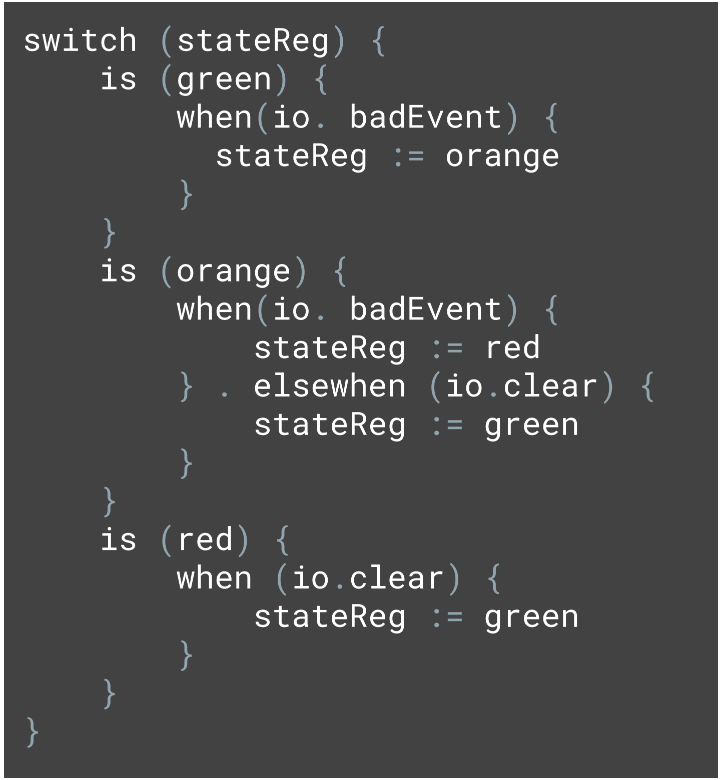
[2,3]

[1] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. V. Lopes, J.-M. Loingtier, and J. Irwin. Aspect-oriented programming. Proceedings of ECOOP '97, 1997.
[2] Eclipse Foundation. Aspectj, 2022. https://www.eclipse.org/aspectj/
[3] Eclipse Foundation. The aspectj programming guide, 2003. https://www.eclipse.org/aspectj/doc/released/progguide/index.html

Vending Machine

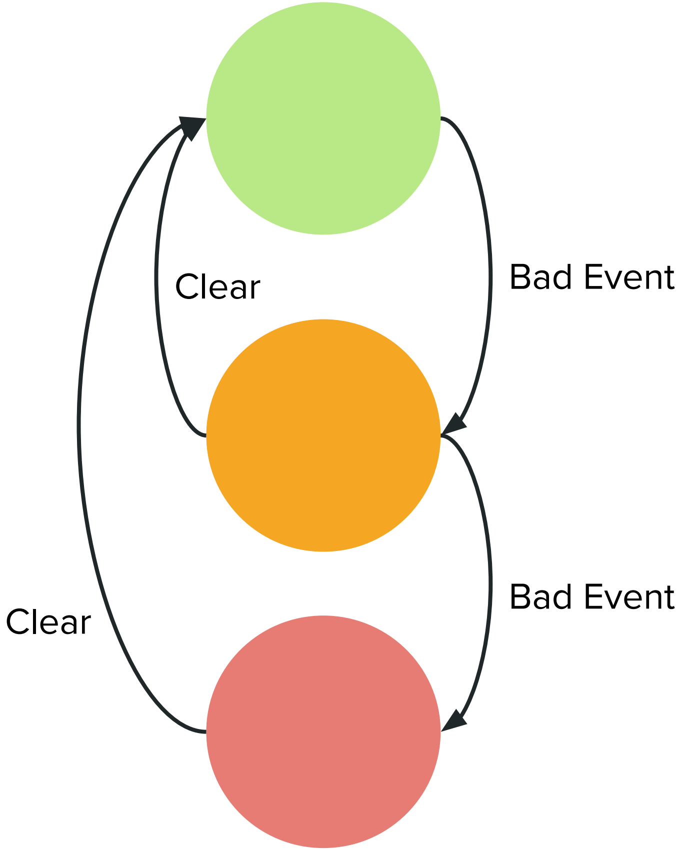


How do we feature orient this?



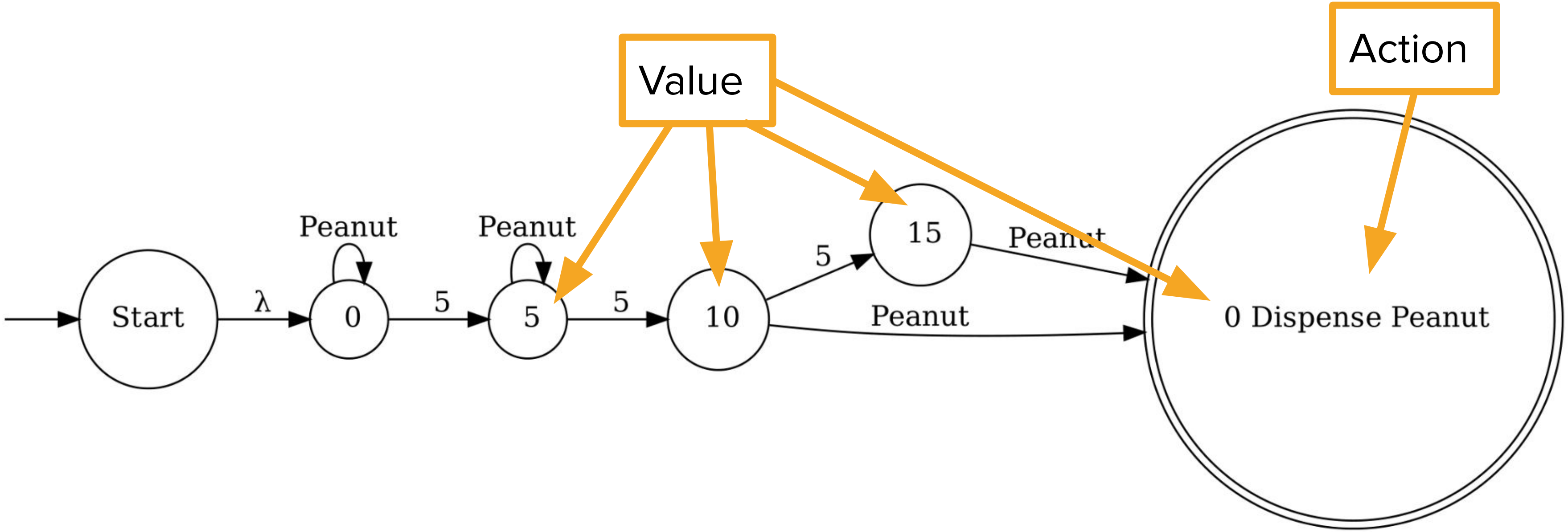
```
switch (stateReg) {
  is (green) {
    when(io. badEvent) {
     stateReg := orange
    }
  }
  is (orange) {
    when(io. badEvent) {
      stateReg := red
    } . elsewhen (io.clear) {
      stateReg := green
    }
  }
  is (red) {
    when (io.clear) {
      stateReg := green
    }
  }
}
```

1

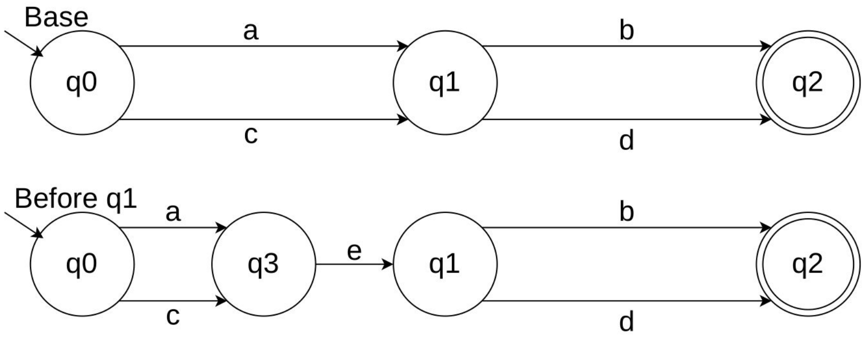


1. M. Schoeberl. Digital Design with Chisel. Kindle Direct Publishing, 2019

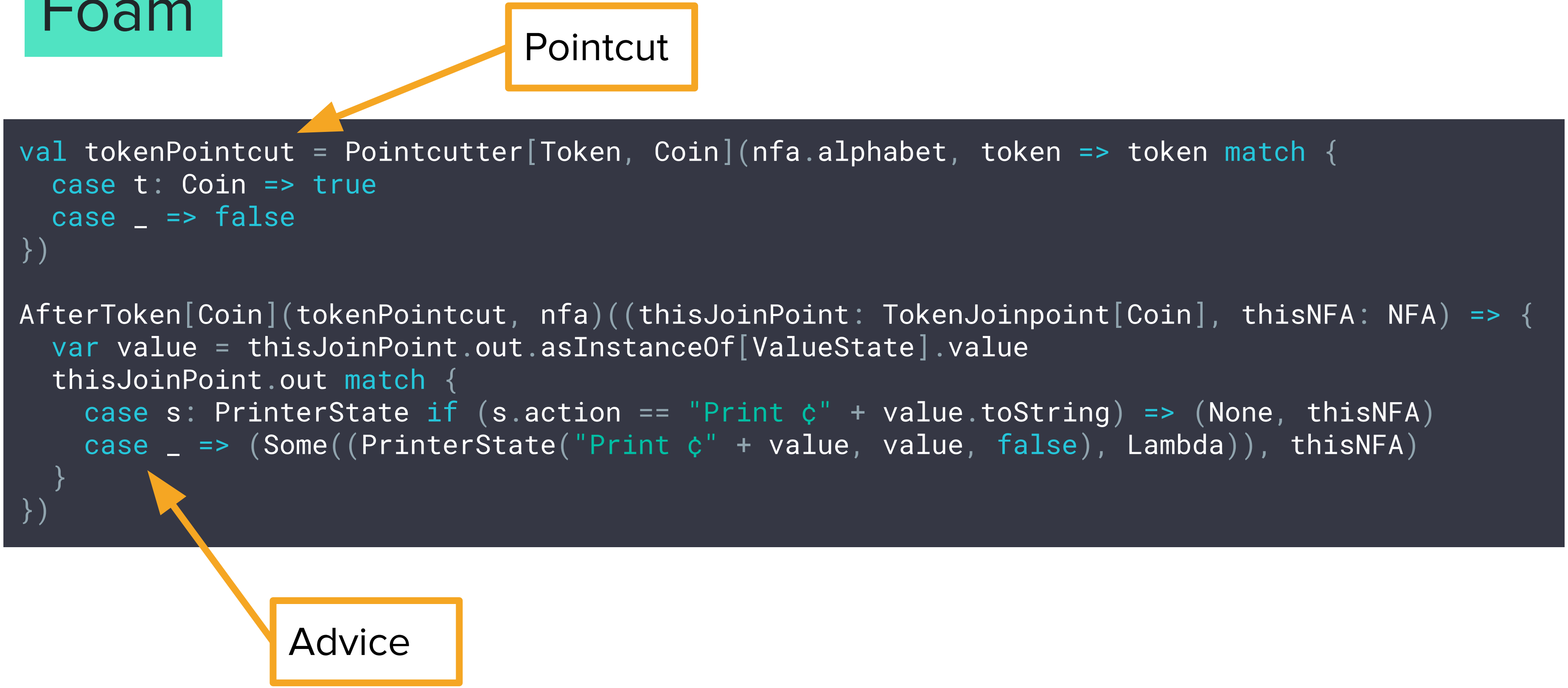
Type Information in FSMs



Advice in FSMs



Foam



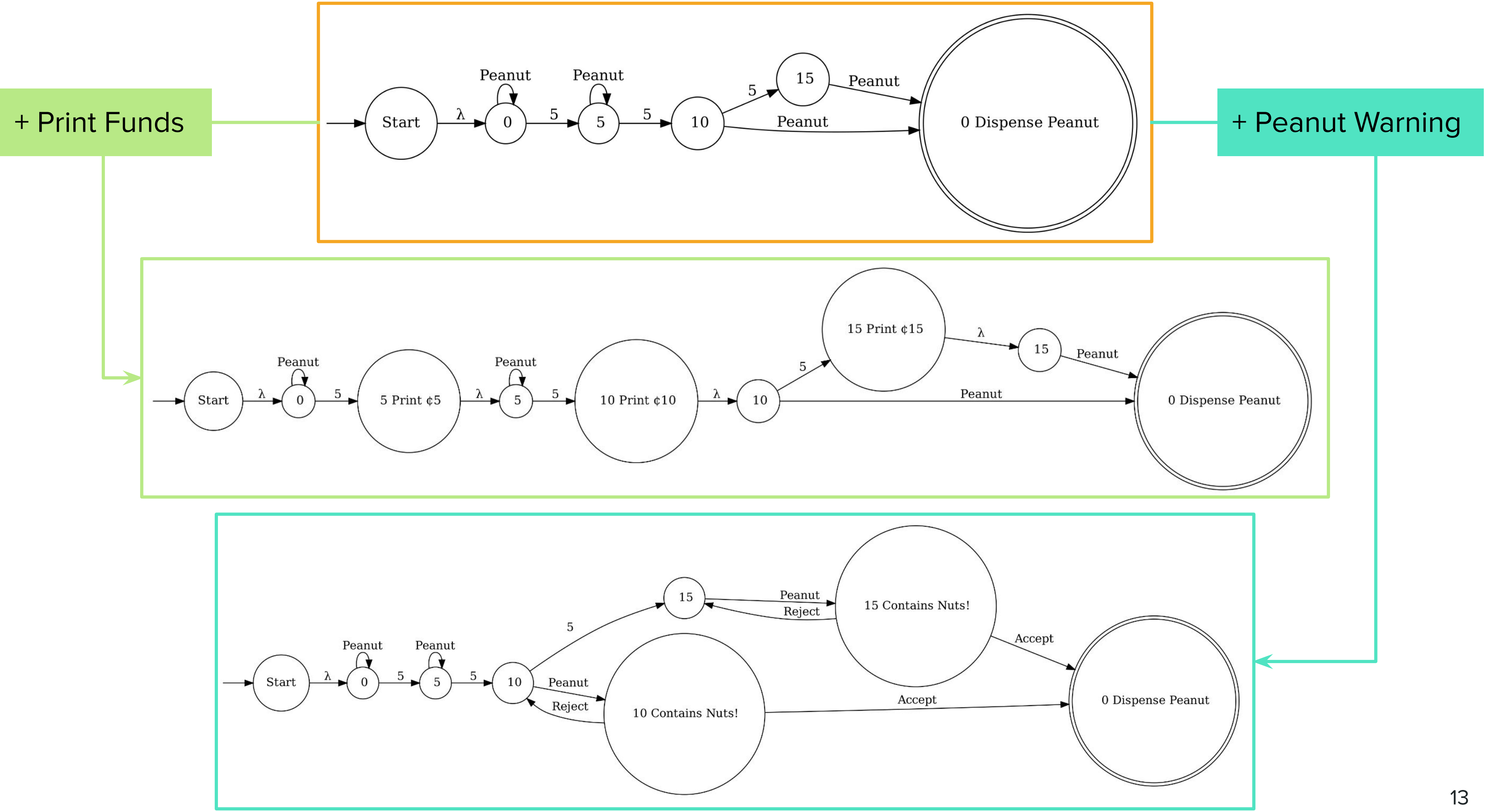
```
val tokenPointcut = Pointcutter[Token, Coin](nfa.alphabet, token => token match {
  case t: Coin => true
  case _ => false
})

AfterToken[Coin](tokenPointcut, nfa)((thisJoinPoint: TokenJoinpoint[Coin], thisNFA: NFA) => {
  var value = thisJoinPoint.out.asInstanceOf[ValueState].value
  thisJoinPoint.out match {
    case s: PrinterState if (s.action == "Print ¢" + value.toString) => (None, thisNFA)
    case _ => (Some((PrinterState("Print ¢" + value, value, false), Lambda)), thisNFA)
  }
})
```

Pointcut

Advice

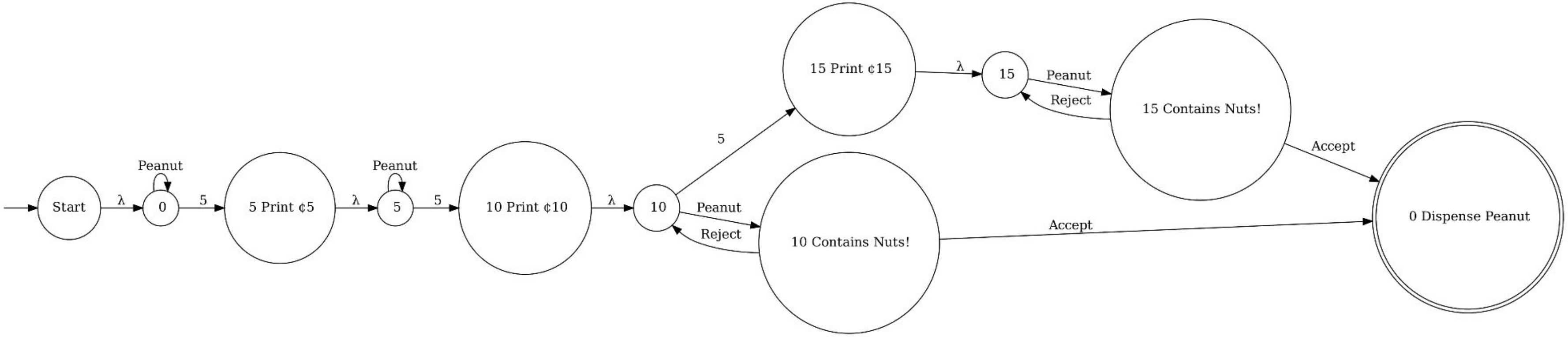
[1] Eclipse Foundation. Aspectj, 2022. https://www.eclipse.org/aspectj/



+ Print Funds

+ Peanut Warning

Vending Machine



Feature Decomposition – Combining Techniques

Cache FSM (Foam Centric)

- Read
- Write
- Acknowledge Idle
- Acknowledge Read
- Dirty Bit Accounting

Cache Hardware (Faust Centric)

- HasBufferBookkeeping
- HasMiddleAllocate
- HasWriteFSM
- HasSimpleWrite
- HasInvalidOnWrite
- HasMiddleUpdate
- HasDirtyBitAccounting
- Dusty

Endpoints

- Endpoints implemented for RISC-V Mini[1]
- All endpoints are fully synthesizable

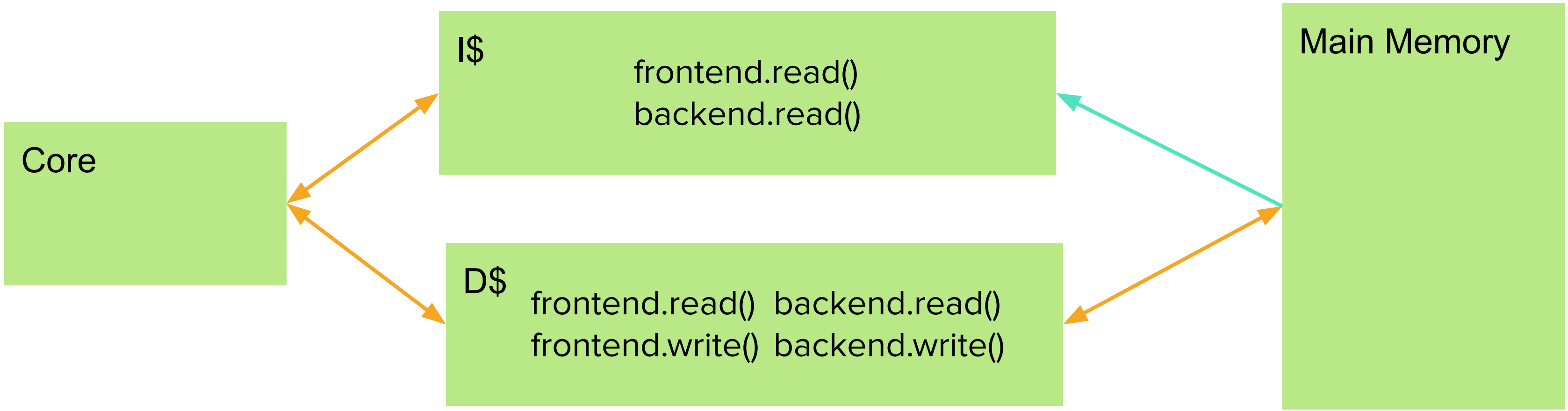
Endpoint	**Features**
Read-Channel	HasWriteStub, HasBufferBookeeping
Read-Only	HasWriteStub, HasMiddleAllocate
Write-Channel	HasWriteFSM, HasSimpleWrite, HasBufferBookeeping, HasInvalidOnWrite
WriteBypass	HasWriteFSM, HasSimpleWrite, HasMiddleAllocate, HasInvalidOnWrite
WriteThrough	HasWriteFSM, HasSimpleWrite, HasMiddleAllocate, HasMiddleUpdate
WriteBack	HasWriteFSM, HasSimpleWrite, HasMiddleAllocate, HasMiddleUpdate, Dirty Accounting
Dusty[2]	HasWriteFSM, HasSimpleWrite, HasMiddleAllocate, HasMiddleUpdate, Dirty Accounting, Dusty

[1] D. Kim. riscv-mini. https://github.com/ucb-bar/riscv-mini, 2022.

[2] S. Friedman, P. Krishnamurthy, R. Chamberlain, R. K. Cytron, and J. E. Fritts. Dusty caches for reference counting garbage collection. In Proc. of Workshop on Memory Performance: Dealing with Applications, Systems and Architecture, Sept. 2005.

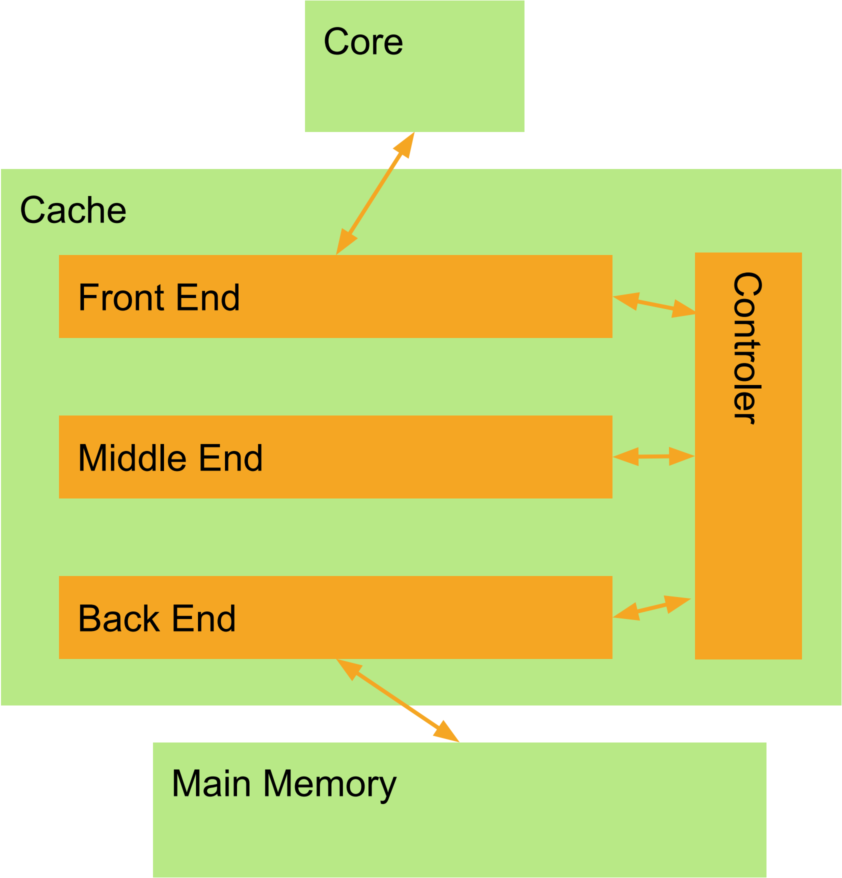
Cache & Types

- Treat the cache as a *type*.
- Generate components by calling methods *only when needed*.
- Use AOP to extend types and insert method calls.



Features as Traits

- Scala Traits allow extending a class *at instantiation.*
- Package features into traits.
- Apply them to *either* the instruction or data cache to add new hardware.
- For features that crosscut types → Use AOP!



Coding Effort

Feature	Chisel	Our Library	Faust	Total
Base System	336	25	0	361
HasWriteStub	10	0	0	10
HasWriteNFA	10	55	0	65
HasSimpleWrite	17	0	0	17
HasBufferBookeeping	35	0	16	51
HasMiddleAllocate	68	0	8	76
HasInvalidOnWrite	12	0	8	20
HasMiddleUpdate	21	0	8	29
Dirty Accounting	11	27	66	104
Dusty	0	0	14	14

Area Measurements

Lower Better

Endpoints	**LUTs (normalized)**
readOnly-dusty	1.81
readOnly-writeBack	1.76
readChannel-dusty	1.57
readOnly-writeThrough	1.56
readChannel-writeBack	1.52
readOnly-writeBypass	1.49
readChannel-writeThrough	1.35
readOnly-writeChannel	1.25
readChannel-writeBypass	1.25
readChannel-writeChannel	1.00

All data normalized to readChannel-writeChannel LUTs.

Performance Measurements

	No Instruction Cache					Instruction Cache				
benchmark	Write Channel	Write Bypass	Write Through	Write Back	Dusty	Write Channel	Write Bypass	Write Through	Write Back	Dusty
median	3.59	3.11	3.11	3.02	2.97	2.41	1.93	1.93	1.87	1.82
multiply	2.81	2.78	2.78	2.77	2.77	1.37	1.33	1.33	1.33	1.33
qsort	3.48	3.38	3.19	3.00	2.99	2.11	2.01	1.83	1.64	1.63
towers	3.78	3.69	3.36	2.46	2.46	2.84	2.76	2.42	1.61	1.61
vvadd	3.71	3.07	3.07	3.00	2.94	2.64	2.00	2.00	1.93	1.87
Average CPI	3.47	3.21	3.10	2.85	2.83	2.27	2.01	1.90	1.68	1.65

Higher CPI → Lower CPI

Conclusion

- Evolve control structures
  - Use AOP to build FSM features
  - Only apply them when needed
- Combine techniques
  - Separate out cache features
  - Combine techniques to evolve the cache and the controller
  - Selectively apply features via rich type information
- Marketplace of features
  - Generalize to whole chip via type system
  - Easily trade features between designers